WHAT IS CLAIMED IS:

1. An application programming interface comprising:

a first interface which controls transfer of information between a first device capable of handling isochronous and asynchronous data and an audio/video file system capable of handling and organizing audio/video data.

2. The application programming interface according to claim 1, further comprising:

a second interface which controls transfer of information between a second device capable of handling asynchronous data and said audio/video file system.

3. The application programming interface according to claim 1, wherein said first device is an audio/video controller.

4. The application programming interface according to claim 3, wherein said audio/video controller is capable of processing commands transmitted using protocol 61883.

5. The application programming interface according to claim 4, wherein said commands are transmitted using protocol 61883 in an isochronous manner.

6. The application programming interface according to claim 2, wherein said second device is a SBP-2 controller.

7. The application programming interface according to claim 6, wherein said SBP-2 controller is capable of processing commands transmitted using serial-bus-protocol-2.

8. The application programming interface according to claim 7, wherein said commands are transmitted using serial-bus-protocol-2 in an asynchronous manner.

9. The application programming interface according to claim 1, wherein control of said transfer of information to and from said first device are independent of internal implementation of said first device.

10. The application programming interface according to claim 2, wherein control of said transfer of information to and from said second device are independent of internal implementation of said second device.

11. The application programming interface according to claim 1 further comprising:

a plurality of function calls.

12. The application programming interface according to claim 11, wherein one or more of said plurality of function calls are designed to allow said audio/video file system to handle a first type of file; and wherein one or more of said plurality of function calls are designed to allow said audio/video file system to handle a second type of file.

13. The application programming interface according to claim 12, wherein said first type of file is a non-audio/video file; and wherein said second type of file is an audio/video file.

14. The application programming interface according to claim 12, wherein said first type of file is smaller than said second type of file.

15. The application programming interface according to claim 11, wherein one or more of said plurality of function calls are designed to allow said audio/video file system to play or record a plurality of audio/video data streams concurrently.

16. The application programming interface according to claim 15, wherein said one or more of said plurality of function calls are designed to allow said audio/video file system to play or record said plurality of audio/video data streams concurrently by using a channel ID parameter and an object ID parameter.

17. The application programming interface according to claim 11, wherein one or more of said plurality of function calls are designed to allow said audio/video file system to play and record an audio/video data stream concurrently.

18. The application programming interface according to claim 17, wherein said one or more of said plurality of function calls are designed to allow said audio/video file system to play and record said audio/video data stream concurrently by using a channel ID parameter and an object ID parameter.

19. The application programming interface according to claim 11, wherein one or more of said plurality of function calls are designed to allow said audio/video file system to initiate a play or record operation starting from within an audio/video file.

20. The application programming interface according to claim 19, wherein said one or more of said plurality of function calls are designed to allow said audio/video file system to initiate a play or record operation starting from within said audio/video file by using an offset parameter.

21. The application programming interface according to claim 11, wherein one or more said plurality of function calls are designed to allow said audio/video file system to optimize disk access.

22. The application programming interface according to claim 21, wherein said one or more of said plurality of function calls are designed to allow said audio/video file system to optimize disk access by designating a first group of function calls to handle a first type of file and a second group of function calls to handle a second type of file.

23. The application programming interface according to claim 22, wherein said first type of file is a non-audio/video file; and wherein said second type of file is an audio/video file.

24. The application programming interface according to claim 11, wherein one or more of said plurality of function calls are designed to allow said audio/video file system to perform a plurality of trick operations with a data stream.

25. The application programming interface according to claim 24, wherein said plurality of trick operations includes a plurality of forward operations.

26. The application programming interface according to claim 25, wherein said plurality of forward operations includes a fast-forward operation, a slow-forward operation, and a step-forward operation.

27. The application programming interface according to claim 24, wherein said plurality of trick operations includes a plurality of reverse operations.

28. The application programming interface according to claim 27, wherein said plurality of reverse operations includes a fast-reverse operation, a slow-reverse operation, and a step-reverse operation.

29. An application programming interface for providing an interface with an audio/video file system capable of handling and organizing audio/video data, comprising:

a first plurality of function calls including:

a load function call designed to cause retrieval of descriptor information from a storage medium;

a store function call designed to cause storing of said descriptor information onto said storage medium;

a delete function call designed to cause deletion of said descriptor information from said storage medium; and

a second plurality of function calls including:

a play function call designed to cause a specified file to be played;

a record function call designed to cause specified data to be recorded; and

a stop function call designed to cause a play or record operation to be stopped.

30. The application programming interface according to claim 29, wherein said first plurality of function calls is designed to handle a first type of file; and wherein said second plurality of function calls is designed to handle a second type of file.

31. The application programming interface according to claim 30, wherein said first type of file is a non-audio/video file; and wherein said second type of file is an audio/video file.

32. The application programming interface according to claim 29, wherein said first plurality of function calls further includes:

a validity function call designed to verify validity of a specified descriptor; and

wherein said second plurality of function calls further includes:

a pause function call designed to cause a play or record operation to be paused;

a resume function call designed to cause a previously paused operation to resume; and

an address retrieval function call designed to determine a logical block address of said specified file during a play or a record operation.

33. The application programming interface according to claim 29, wherein said second plurality of function calls includes:

a plurality of function calls designed to cause forward operations to be performed; and

a plurality of function calls designed to cause reverse operations to be performed.

34. The application programming interface according to claim 33, wherein said plurality of function calls designed to cause forward operations to be performed includes:

a fast-forward function call;

a slow-forward function call; and

a step-forward function call.

35. The application programming interface according to claim 33, wherein said plurality of function calls designed to cause reverse operations to be performed includes:

a fast-reverse function call;

a slow-reverse function call; and

a step-reverse function call.

36. The application programming interface according to claim 29, wherein said application programming interface is capable of being used by a first device

capable of handling isochronous and asynchronous data to communicate with said audio/video file system.

37. The application programming interface according to claim 36, wherein said first device is an audio/video controller.

38. The application programming interface according to claim 36, wherein said application programming interface is capable of being used by a second device capable of handling asynchronous data to communicate with said audio/video file system.

39. The application programming interface according to claim 38; wherein said first device is a SBP-2 controller.

40. The application programming interface according to claim 32, wherein said specified descriptor is an object descriptor.

41. The application programming interface according to claim 32, wherein said specified descriptor is a content list.

42. The application programming interface according to claim 32, wherein said specified descriptor is a performance list.

43. The application programming interface according to claim 32, wherein said specified descriptor is a HMS table.

44. The application programming interface according to claim 32, wherein each of said first and second plurality of function calls is capable of passing a plurality of parameters.

45. The application programming interface according to claim 44, wherein said plurality of parameters that is capable of being passed by said load function call includes a descriptor ID parameter, a type parameter, an offset parameter, a size parameter, a data_location parameter, and a call_back parameter.

46. The application programming interface according to claim 44, wherein said plurality of parameters that is capable of being passed by said store function

call includes a descriptor ID parameter, a type parameter, an offset parameter, a size parameter, a data_location parameter, and a call_back parameter.

47. The application programming interface according to claim 44, wherein said plurality of parameters that is capable of being passed by said delete function call includes a descriptor ID parameter, a type parameter, and a call_back parameter.

48. The application programming interface according to claim 44, wherein said plurality of parameters that is capable of being passed by said play function call includes a channel ID parameter, an object ID parameter, a start_position parameter, an_end position parameter, a speed parameter, and a call_back parameter.

49. The application programming interface according to claim 44, wherein said plurality of parameters that are capable of being passed by said record function call include a channel ID parameter, an object ID parameter, a start_position parameter, a type parameter, and a call_back parameter.

50. The application programming interface according to claim 44, wherein said plurality of parameters that is capable of being passed by said stop function call includes a channel ID parameter, a call_back parameter, and a logical_byte_address parameter.

51. The application programming interface according to claim 44, wherein said plurality of parameters that is capable of being passed by said pause function call includes a channel ID parameter, a call_back parameter, and a logical_byte_address parameter.

52. The application programming interface according to claim 44, wherein said plurality of parameters that is capable of being passed by said resume function call includes a channel ID parameter and a call_back parameter.

53. The application programming interface according to claim 44, wherein said plurality of parameters that is capable of being passed by said address retrieval function call includes a channel ID parameter and a count parameter.

54. The application programming interface according to claim 44, wherein said plurality of parameters that is capable of being passed by said validity function call includes a descriptor ID parameter, a type parameter and a call_back parameter.

55. The application programming interface according to claim 34, wherein said fast-forward function call is capable of passing a plurality of parameters including a channel ID parameter, a type parameter, an interval parameter, a repeat parameter, and a call_back parameter.

56. The application programming interface according to claim 34, wherein said slow-forward function call is capable of passing a plurality of parameters including a channel ID parameter, a repeat parameter, an increment parameter and a call_back parameter.

57. The application programming interface according to claim 34, wherein said step-forward function call is capable of passing a plurality of parameters including a channel ID parameter, an increment parameter and a call_back parameter.

58. The application programming interface according to claim 35, wherein said fast-reverse function call is capable of passing a plurality of parameters including a channel ID parameter, a type parameter, an interval parameter, a repeat parameter, and a call_back parameter.

59. The application programming interface according to claim 35, wherein said slow-reverse function call is capable of passing a plurality of parameters including a channel ID parameter, a repeat parameter, an increment parameter and a call_back parameter.

60. The application programming interface according to claim 35, wherein said step-reverse function call is capable of passing a plurality of parameters including a channel ID parameter, an increment parameter and a call_back parameter.

61. A method for providing communication with an audio/video file system, comprising steps of:

providing a first interface which controls transfers of information between said audio/video system and a first device capable of handling isochronous and asynchronous data; and

providing a second interface which controls transfers of information between said audio/video system and a second device capable of handling asynchronous data.

62. The method according to claim 61, wherein said signals transferred between said audio/video system and said first device are independent of internal implementation of said first device; and

wherein said signals transferred between said audio/video system and said second device are independent of internal implementation of said second device.